RECEIVED
CENTRAL FAX CENTER
APR 23 2008

AMENDMENTS TO THE CLAIMS

This listing of claims will replace all prior versions, and listings, of claims in the application.

1-20 (Cancelled)

21. (Currently Amended) A method comprising:

determining whether an instruction for a first thread is of a first type;

pausing ~~processing~~ execution of instructions of the first thread upon determining that the instruction is of the first type while executing ~~processing~~ instructions from a second thread; and

resuming execution ~~processing~~ of a second microinstruction ~~instruction~~ decoded from the instruction after ~~execution~~ retirement of a first microinstruction ~~instruction~~ decoded from the instruction.

22. (Currently Amended) The method of claim 21 further comprising decoding the instruction into the first microinstruction ~~instruction~~ and the second microinstruction ~~instruction~~.

23. (Currently Amended) The method of claim 22 wherein the first microinstruction ~~instruction~~ causes a value to be stored in memory for the first thread.

24. (Currently Amended) The method of claim 23 further comprising:

processing the second microinstruction ~~instruction~~ for execution when the value stored in the memory is reset.

25. (Currently Amended) The method of claim 24 wherein the value stored in the memory is reset when the first microinstruction ~~instruction~~ is retired.

26. (Currently Amended) A method comprising:

determining whether an instruction of a first thread is of a first type;

initiating a counter upon determining that the instruction of the first thread is of the first type, wherein the instruction of the first thread includes an operand and the initiating includes loading the counter with a value of the operand; and

pausing ~~processing~~ execution of instructions of the first thread until the counter reaches the value of the operand while ~~processing~~ executing instructions of a second thread at [[the]] a pipeline stage.

27. (Cancelled)

28. (Currently Amended) The method of claim 26 further comprising resuming executing ~~processing~~ instructions of the first thread after the counter reaches the value of the operand.

29. (Currently Amended) An apparatus, comprising:

a decode unit to determine whether an instruction of a first thread is of a first type, said decode unit to pause ~~processing~~ execution of instructions of said first thread upon determining that the instruction of the first thread is of the first type by generating a first microinstruction ~~instruction~~ to cause a value to be stored in a memory for the first thread while instructions from a second thread can be executed ~~processed~~, said decode unit further to generate a second microinstruction ~~instruction~~ upon which execution ~~processing~~ is to resume after retirement ~~execution~~ of the first microinstruction ~~instruction~~.

30. (Cancelled)

31. (Cancelled)

32. (Currently Amended) The apparatus of claim 29 wherein the decode unit is to allow

~~process~~ the second microinstruction ~~instruction~~ to be executed when the value stored in the memory is reset.

33. (Currently Amended) The apparatus of claim 32 further comprising:

a retire unit coupled to the decode unit, wherein the retire unit is to cause the value stored in the memory to be reset when the first microinstruction ~~instruction~~ is retired by the retire unit.

34. (Currently Amended) An apparatus comprising:

a decode unit to determine whether a first instruction for a first thread is of a first type;

a counter coupled to the decode unit, the counter to be loaded with a value of an operand of the first instruction if the first instruction for the first thread is of the first type, the decode unit to pause ~~processing~~ executing instructions of the first thread until the counter reaches [[a]] the value of the operand; and

wherein instructions for a second thread can be ~~processed~~ executed while the instructions of the first thread are paused from being ~~processed~~ executed.

35. (Cancelled)

36. (Currently Amended) The apparatus of claim 34 wherein the decode unit is to operate while the first thread is paused from being executed ~~processed~~.

37. (Previously Presented) The apparatus of claim 29, implemented in a system including operating system code to cause the instruction to be forward to the decode unit.

38. (Currently Amended) The apparatus of claim 34, implemented in a system including operating system code to cause the first instruction to be forward to the decode unit.

39. (Currently Amended) An apparatus comprising:

a queue; and

a decode unit coupled with the queue to receive and decode a pause instruction of a first thread, the decode unit in response to the pause instruction to pause execution ~~processing~~ of instructions of the first thread for a period of time while instructions from a second thread are executed ~~processed~~ and the decode unit in response to the pause instruction to generate a microinstruction ~~an instruction~~ that is to be queued in the queue during the period of time, and the decode unit further to cause resumption of execution of instructions of the first thread after the period of time.

40. (Currently Amended) An apparatus comprising:

a decode unit to receive and decode a pause instruction of a first thread, the decode unit in response to the pause instruction to pause execution ~~processing~~ of instructions of the first thread while instructions from a second thread are executed ~~processed~~ by generating a first instruction and a second instruction, the first instruction to set a flag before being forwarded to an execution unit and the second instruction to prevent the instructions of the first thread from being executed ~~processed~~ until after the flag has been reset after retirement of the first instruction ~~resets the flag~~.

41. (Currently Amended) A pause instruction stored in one of a memory and a queue that is operable to cause a processor to perform operations comprising:

decoding the pause instruction;

loading a value of an operand of the pause instruction into a counter; [[and]]

pausing ~~processing~~ execution of instructions of a first thread from which the pause instruction was received while instructions of a second thread are executed ~~processed~~ until the counter reaches the value of the operand; and

resuming execution of instructions of the first thread after the counter reaches the value of the operand.